Cooperation Strategies in a Time-Stepped Simulation of Foraging Robots

Liam McGuigan, Catherine Saunders, Roy Sterritt, George Wilkie
School of Computing, Faculty of Computing, Engineering and the Built Environment
Ulster University
Jordanstown, N. Ireland
email: mcguigan-l8@ulster.ac.uk, c.saunders@ulster.ac.uk, r.sterritt@ulster.ac.uk, fg.wilkie@ulster.ac.uk

***Abstract*—Large robotic swarms may be used to carry out tasks, such as space exploration, mining, search & rescue operations and more. To enable their use in these fields, the individual robots within a swarm will need to be autonomic, capable of making their own decisions and adjusting their behaviour without relying on regular human intervention. This paper demonstrates the potential for autonomic self-adaptation within a swarm of foraging robots by investigating the performance of different cooperation strategies in different scenarios. The results show that the performances of the strategies are affected by operational conditions that can change over the course of a mission, and that the autonomic capability to self-adapt would prove beneficial. Additionally, the time-stepped simulation used here is compared to the performance of a previous approach using real-time simulation, with a view to identifying which approach is more suitable for embedding within a robot as a means of aiding that autonomic process through simulating potential options. The time-stepped simulation is found to be faster and more efficient, and therefore more suited to embedding.**

Keywords- Swarm robotics; Self-adaptation; Autonomic Computing; Simulation.

I. INTRODUCTION

The use of robotic swarms consisting of a large number of robots operating in concert can benefit applications, such as space exploration [1][2], search & rescue [3] and mine clearance [3][4] among others, taking advantage of a robot's ability to operate in conditions where human involvement is too dangerous or difficult.

The individual craft in a robotic swarm will need to be capable of managing themselves without requiring constant supervision. They may be required to make quick decisions to protect themselves or to act on opportunities, and will need to adapt to best suit the conditions of the task being carried out [5]. This can be achieved by making the swarms autonomic.

Autonomic computing concepts will embody the swarm with the properties of self-configuration, self-healing, self-optimization and self-protection, ensuring that the swarm [6] is implemented by including an autonomic component running a Monitor, Analyse, Plan, Execute, with a shared Knowledge (MAPE-K) control loop to monitor and analyse the situation, and plan and execute any changes to behaviour aided by a knowledge base of pre-set or previously acquired information [7]. Autonomic robotics combines the concepts of MAPE-K from autonomic computing, with Intelligent Machine Design (IMD) from robotics [8][9].

Due to the cost and impracticality of using real hardware during the development of large-scale swarm behaviour with real hardware, simulators are often used in the process [10], able to create artificial swarms of hundreds or even thousands of robots engaged in tasks, such as foraging, surveillance and exploration of unknown environments.

The research presented in this paper has two objectives. The first is to investigate the potential for self-adaptation through selection of a cooperation strategy during a foraging task, through analysis of the performance of three different strategies over the course of the task. The second objective is to identify which of two simulation approaches used would be most suitable for deploying on an individual agent within the swarm as a means of using simulation-in-the-loop to help with the decision to switch.

The rest of this paper is organised as follows. Section II discusses related work on self-adaptation in swarm robotics, and the varying use of simulations in development. Section III describes the cooperation strategies developed in a previous study [11], and the implementation of an alternative simulation for exploring them, setting out the test scenarios to be run. Section IV presents these results and the implications, including a comparison of the two simulation approaches used. Section V concludes the paper with a summary and indicates the future research directions.

II. RELATED WORK

The following subsections discuss current research in swarm self-adaptation, and the use of simulations within swarm development.

A. Swarm Self-Adaptation

Self-adaptation of a robotic swarm concerns the ability of the swarm to adjust its behaviour in response to external or internal conditions, such as a foraging swarm choosing to abandon a depleted deposit in order to find newer deposits, or a surveillance swarm organizing itself so as to provide maximum coverage of the target area.

Swarm self-adaptation can be considered based on two factors – the approach to adaptation, and the location within the swarm where this is applied. Approaches to swarm adaptation include engineering emergent processes where adaptation arises naturally out of the agent behaviour [12], reasoning and learning approaches where the swarm explicitly reasons about the decision being made [13] and

 ISBN: 978-1-61208-781-8

may learn from experience [14], and evolutionary approaches which explore alternatives through genetic algorithms [15].

Regarding location, a lot of the research focuses on applying adaptation to individual agent behaviour [16]–[18]. This low-level adaptation results in a bottom-up approach to swarm behaviour, with the resulting performance of the swarm arising from the aggregate performance of the individual agents. This can allow for more specific adaptation, such as balancing an individual's conflicting objectives [19], which may be difficult to apply at the swarm level. Agent behaviour adaptation can have the most direct impact on the swarm's performance, but it is difficult for an agent to make an individual decision on aspects of collaboration or coordination between multiple agents.

Adaptation through the selection of swarm-level cooperation strategies can be used to address the problem of collaboration. In this approach, agents within the swarm can collectively determine an alternative approach which is swapped with the existing agent behaviour either in part or in whole. This selection may be driven by an autonomic component that assesses the suitability of alternative strategies [11][14], and may be employed with in a subset of the entire swarm [20].

This research is focused on identifying the potential for swarm-level adaptational changes by assessing the performance of a selection of candidate strategies in a set of scenarios. Through noting any effect the scenario has on the performance of a particular strategy, the benefits of the ability to select an alternative strategy will become apparent.

B. *Use of Simulations in Swarm Development*

Simulation has long been employed as a tool for the development of robotic and swarm simulations, providing the means to test and analyse systems in an artificial environment. Simulators range in complexity, from detailed physical simulations of actual robots [10][21], to abstract approaches where robots move within a grid-based environment. The difficulty of producing an accurate simulation of the real world can manifest as the "reality gap" [22], where results obtained in simulation are not replicated in reality. Nonetheless, it is not necessary for the results of a simulation to be precisely reproducible in the real world for the simulation itself to prove useful.

Simulation need not be restricted to the offline development phase. It may be used to assist the decision making process [23][24], trying out "what-if" scenarios in order to assess the effects of potential actions or strategies ahead of time. For this to be effective, a simulation must be detailed enough to provide useful information, while remaining lightweight enough to be able to run on an individual agent within the swarm.

When choosing or designing a simulator for researching robotic swarms, the accuracy of the physical simulation required will depend on the impact specific hardware has on the research being conducted. Developing a robotic controller without a suitably accurate physical simulation can lead to the robots in the simulation carrying out behaviour that is impossible with the actual robots [25], but when researching purely software based systems, abstractions can be used to trade accuracy for a faster execution time [21].

Further gains in execution time may be made by simplifying the world representation. Grid based approaches need not produce markedly different results to continuous space [26], and can be used in cases where the specific motion of agents can be abstracted.

The majority of multi-robot simulators available make use of discrete time when updating their simulations, in which all agents and physical reactions are updated in sequence with a small time step, rather than independent execution in real time, such as assigning a robot its own execution thread. This ensures synchronous execution of robots [21] and simplifies physical interactions.

The research conducted in this paper abstracts physical movement using cell-based movement within a grid, and the performance is contrasted with a real-time, multi-threaded approach used in previous research [11].

III. TIME-STEPPED SIMULATION OF COOPERATION STRATEGIES

This research makes use of a simulation of a heterogeneous swarm of agents engaged in a variant of a foraging task. The 30x30 world is seeded with 100 each of two different types of items, and 200 robots are divided into two equal groups based on which item they can process. Items are processed in-situ, rather than returned to a home base – the process is analogous to applications, such as mine deactivation, analysis of mineral deposits or environmental clean-up. The task is considered complete when all items have been successfully foraged.

The world is represented as a grid of equal sized cells, one unit square, with items and robots each occupying a single cell. During world generation, items are placed so as not to share cells, but there is no restriction on robots sharing a single cell. Figure 1 shows the world at the beginning of a simulation.

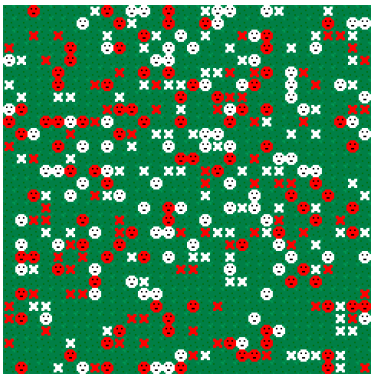


Figure 1. Initial setup of the simulation. The colour of a robot (face) or item (X) indicates its type.

A. Previous Implementation

The performance of the time-stepped simulation presented here will be compared with that of the threaded implementation in the previous work [9].

In the threaded implementation, each robot was run on an independent CPU thread, with a reliance on real-time delays when messages required a response, as in the One Responder strategy. Each robot's progress was also artificially delayed in order to allow the task to be viewed within the simulation program. The updating of the individual robots is thus subject to the CPU's thread scheduling, and cannot be predicted.

This reliance on real-time delays is not present in the new time-stepped approach where a unit of time is defined by a single tick of the simulation during which each robot is updated in sequence.

B. Cooperation Strategies

Cooperation during the task is determined by the use of one of three strategies, as developed in [11]. When a robot encounters an item that it cannot forage, it broadcasts a help request with a range of 5 units. The behaviour of the robots is then determined by the strategy:

1) Multiple Responders: A receiving robot, if not already engaged in foraging or responding to a previous request, will respond to the request by moving towards the item if it is able to forage it. All receivers, whether they can forage the item or not, will rebroadcast the message. In this way, the message will filter throughout the swarm. The robot initiating the help request plays no further part in the cooperation and returns to exploration.

2) Selective Responders: Behaviour is similar to the Multiple Responders approach, but the message is only rebroadcast if the receiving robot cannot help. This works to reduce the number of robots responding to the request.

3) One Responder: The robot initiating the request waits for offers of help, which are sent by receiving robots that meet the criteria. No rebroadcasting of the message occurs. If no offers are received after a short delay, the requesting robot returns to its previous behaviour, otherwise it assigns the task to the nearest responding robot and resumes exploration. Robots that do not receive assignment after a period of time return to exploration.

Both Multiple and Selective strategies are likely to result in multiple robots moving towards the item. This would provide contingency in the event of robot failure before reaching the target item. Robot failure is not implemented in the current simulation, but will be in a future study.

C. Robot Behaviour

The behaviour of each robot is controlled using a finite state machine (FSM). Figure 2 shows a simplified diagram of the transitions between the three states used for the Multiple and Selective Responder strategies.

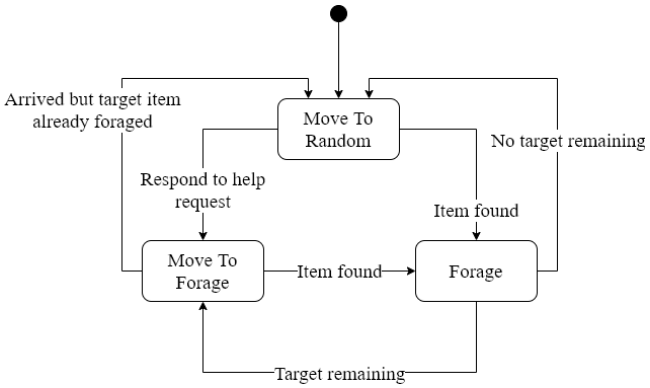


Figure 2. States and transitions used for the Multiple and Selective Responder strategies.

A robot begins in the MoveToRandom state, where it will select a random location in the world and move towards it. Each step, if an item is found, the robot will transition to the Forage state.

In the Forage state, the robot checks if the item is the right type, and forages successfully if so. If not, it will broadcast for help according to the selected strategy. It will then return to its previous state based on whether it is answering a request of its own.

A robot responding to a help request transitions to the MoveToForage state, which is similar to MoveToRandom except the destination is that of the item for which help was requested. To prevent robots from being distracted by new requests, help requests are only processed by a robot in the MoveToRandom state.

When using the One Responder strategy, two additional states are used. A robot broadcasting for help transitions to the WaitForHelpOffers state for two simulation ticks, before selecting the nearest robot. Robots that respond to the initial request transition to the WaitForHelpAssignment state for three ticks before returning to their previous task if not selected.

If no cooperation strategy is used, robots transition between only the MoveToRandom and Forage states.

D. Simulated Messages

In a time-stepped simulation, the potential effects of the agent update order need to be managed. For example, during a single update, Agent 2 may broadcast a message that would be received by Agents 1 and 3. If those values also represent their update order, Agent 3 would be able to receive and respond to the message in the same update, while Agent 1 would need to wait until the following update. This is not desirable behaviour.

To avoid this, messages sent by a robot during an update are queued by the simulation and sent out at the end of the update. If an agent that receives a message in turn broadcasts one of its own, this will not be sent out until the next update. In this way, robots are unable to instantaneously receive responses without any time delay.

 ISBN: 978-1-61208-781-8

E. Experimental Scenarios

The following scenarios were created in order to test the effects of each cooperation strategy in different situations.

1) Equal Split of Items and Robots: 100 red robots and 100 white robots attempt to locate and forage 100 red items and 100 white items.

2) Robot Type Imbalance: 180 red robots are used and only 20 white robots, while the items are equally distributed, representing a scenario where the swarm configuration deployed is not best suited to the task and must adapt.

3) Item Type Imbalance: 180 red items and 20 white items, with equal robot distribution, representing a scenario where the reality of the mission differs from the expected, and again the swarm must adapt.

Each scenario is tested with two map sizes, 30x30 and 90x90, with the latter used to test performance in less concentrated environments. Each simulation is run 30 times, with the initial placement of items and robots randomised at the start of each run. For the threaded simulation, the Equal and Robot Imbalance scenarios were each run 10 times on the 30x30 map only due to simulation limitations, also with randomisation of item and robot placement.

In assessing the performance of each strategy, the number of simulation ticks until completion of the task is the main metric, as it is a measure of the time taken to forage all items. If the energy cost of actions taken by robots is of interest, then the total number of steps and the number of messages broadcast will also become factors. The simulation does not currently assign an energy cost to individual actions, but the counts may be used as a guide, and for each metric a lower value is considered more efficient.

IV. RESULTS

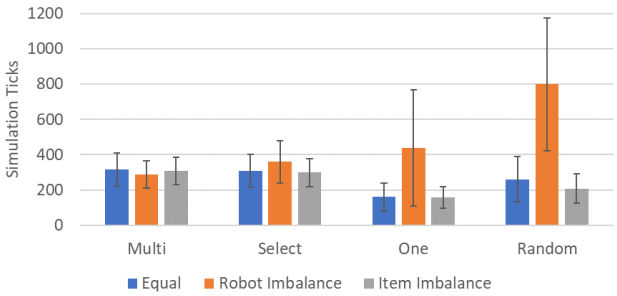
The following subsections compare the performance of the cooperation strategies in the tested scenarios, followed by a comparison of the two simulation methods employed.

A. Cooperation Strategies

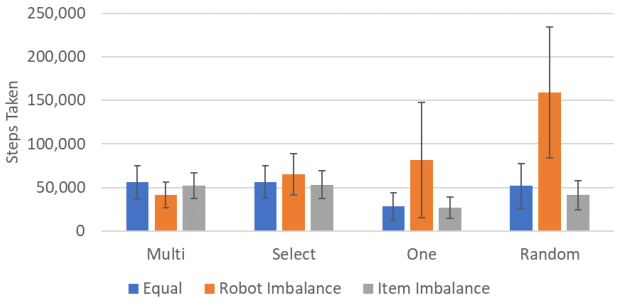
Figure 3 shows the ticks to completion, steps taken, and messages sent for each of the test scenarios in a 30x30 grid.

Comparing the results in both the Equal (a) and Item Imbalance (c) strategies, the One Responder strategy is the best performing approach, having the lowest count in each metric. Multiple and Selective Responder strategies can actually perform worse than no cooperation strategy at all, which can be explained by robots that respond to messages halting any exploration while they respond.

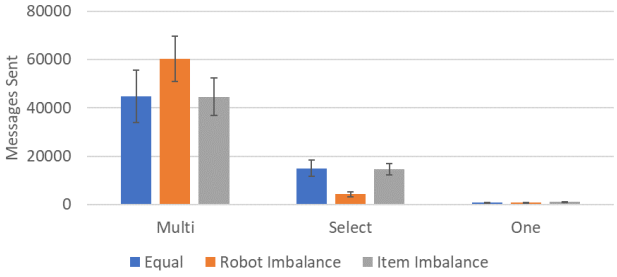
In the Robot Imbalance scenario (Figure 3 (b)), however, One Responder does not reliably perform, and is subject to a great deal of variance caused by the initial placement of items and robots, and the subsequent movement of robots within the arena.



(a) Simulation ticks



(b) Steps taken



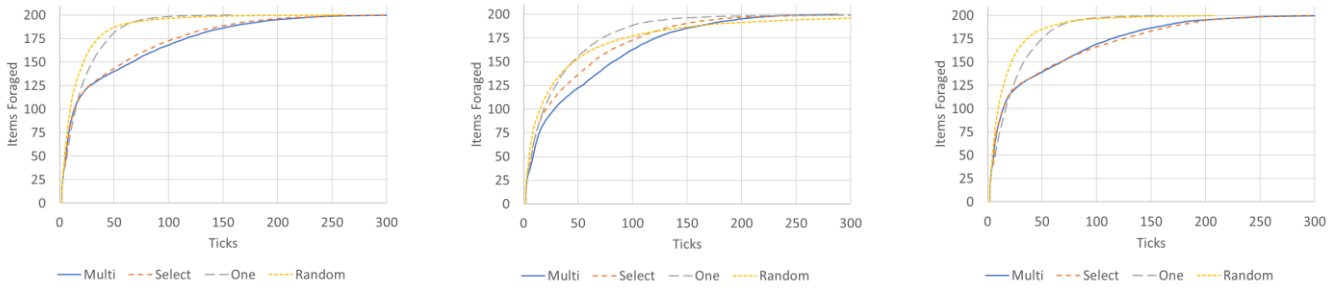
(c) Messages sent

Figure 3. Metrics for each cooperation strategy in a 30x30 map. Error bars represent one standard deviation: (a) simulation ticks, (b) total steps taken, (c) messages sent.

When considering energy costs, Multiple Responders has an extremely high message count setting it apart from Selective Responders, which it otherwise performs very similarly to. A full assessment of the respective efficiency of each would require an assignment of cost to each of the metrics, with the total cost calculated accordingly.

Figure 4 shows the progress of each strategy over time for the three scenarios. In Equal (a) and Item Imbalance (c) scenarios, performance is again similar, however it is notable that using no cooperation strategy at all is the quickest approach until the item count decreases substantially, after which One Responder performs best. This would suggest some system of changing the cooperation strategy used during the test based on the changing situation could lead to stronger overall performance, at least in terms of time taken.

 ISBN: 978-1-61208-781-8



(a) Equal (b) Robot imbalance (c) Item imbalance

Figure 4. Items foraged over simulation ticks for each of the strategies in a 30x30 map: (a) Equal scenario, (b) Robot imbalance scenario, (c) Item imbalance scenario.

In Figure 4 (b), the Robot Imbalance scenario shows only a slight favouring of Random and One Responder scenarios until most items are gathered, but the imbalance of robots then leads to both strategies taking much longer to complete the task than the other approaches. Again, strategy selection during the task could recognise this situation and adopt the strategy most suited.

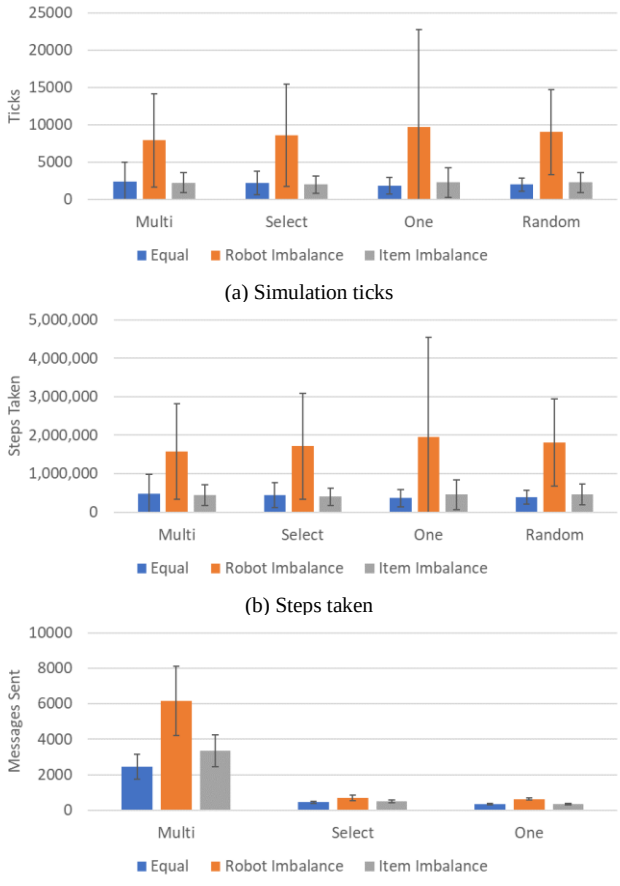
If individual robot failure is considered, a robot imbalance can occur during the task. A system that can monitor the current swarm composition as well as estimate the progress in the task would therefore be able to adopt a suitable strategy in response to such unpredictable change.

Figure 5 shows the ticks to complete, steps taken, and messages sent for the cooperation strategies in the larger 90x90 grid. Here, it can be seen that the performance of each strategy tends towards that of no cooperation, with large variances in the data and, other than the number of messages sent, similar average values for each metric in each scenario.

It may be expected that the larger map explains the results as messages are not being broadcast far enough in order be received, but a comparison of data in Figure 6 shows that this is not necessarily the case. The proportion of requests receiving a response does not change much between the map sizes for the Multiple and Selective Responders cases, other than when there is a robot imbalance where it can be understood the chances of a robot of the correct type being nearby is significantly lower in a larger area.

The One Responder strategy can be seen to have a much higher percentage of requests receiving a response than the other approaches in a 30x30 map. This is due to the other approaches causing robots who would be able to help to be otherwise engaged in moving to forage an item, and thus unable to respond until they complete that help task. As the One Responder strategy causes only one robot at most to take on a task, other robots remain to be selected. In the 90x90 map, this then drops because of the distance between robots, and more closely matches the other approaches.

The dominant effect in the 90x90 map is the random exploration of the environment, and can be seen in the time taken to complete the task and understood by considering that the number of items remains the same between the two maps. As such, only 2.5% of the cells in a 90x90 map have an item, compared to 22.2% of the cells in the 30x30 map. It is this decreased chance of stumbling upon an item that has the strongest effect on swarm performance.

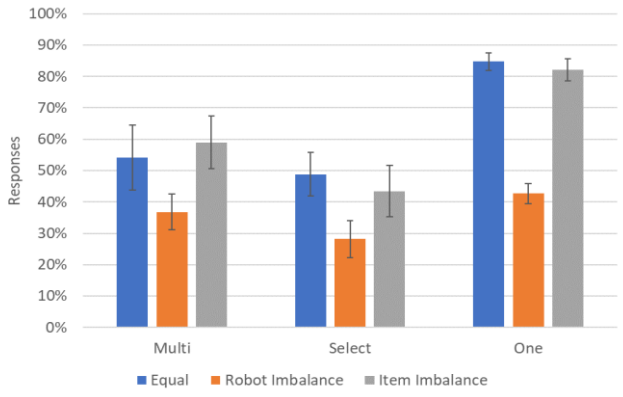


(a) Simulation ticks

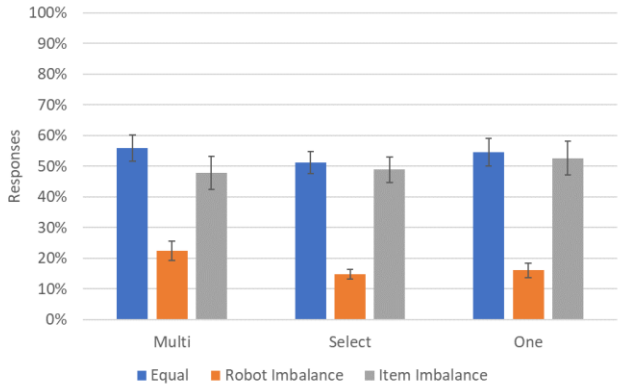
(b) Steps taken

(c) Messages sent

Figure 5. Metrics for each cooperation strategy in a 90x90 map. Error bars represent one standard deviation: (a) simulation ticks, (b) total steps taken, (c) messages sent.



(a) 30x30 map



(b) 90x90 map

Figure 6. Percentage of help requests receiving at least one response, for each cooperaton strategy and scenario. Error bars represent one standard deviation: (a) 30x30 map, (b) 90x90 map.

The results show that allowing a swarm to adjust its cooperation strategy during a task, rather than relying on an initial strategy, could prove beneficial to performance by allowing the swarm to adjust its approach in response to the situation. This self-adaptation applied at the swarm strategy level would require the swarm to have knowledge of its own composition, the current state of the task, and environmental factors, as well as the ability to carry out the analysis of the situation in a decentralised fashion.

B. Simulator Comparisons

Table I compares the time taken to complete the task for each of the simulators in the Equal scenario. The time-stepped simulation presented here is significantly faster than the previous simulation. This can largely be accounted for by the deliberate delays introduced previously to allow for visualization, with some impact of the reliance on real-time delays for communication, which makes a true comparison difficult.

TABLE I. SIMULATON DURATION (EQUAL SCENARIO)

	Multi time (s)	Select time (s)	One time (s)
Time-Stepped	2.11	0.89	0.17
Threaded	155.14	142.01	130.86

Figure 7 shows that the time-stepped simulation takes a much larger number of steps in the Multiple and Selective Responder strategies, and also shows an increase under One Responder. This unexpected result may be explained by the specific behaviour of the robots in each simulation. In the threaded approach, robots pause frequently, the effect of which is that fewer robots will move in each step. For example, on deciding to respond to a help request a robot pauses for three seconds. Further, if another help request is received during that pause, that too may be processed and the robot may choose to act on that, with a further pause.

The effect of these pauses is to reduce the number of robots moving at any given time. In the time-stepped simulation, a robot will only pause when waiting for help responses or assignments in the One Responder strategy.

It is notable that despite these pauses, robots in the threaded approach take fewer steps overall, rather than taking the same number of steps over a longer period. This suggests there may be a benefit to reducing the number of robots engaged in random exploration, but this will need to be investigated further.

The impact of the two simulations on the host platform was compared and Table II displays the approximate processor and memory usage of the two platforms when running simulations.

Overall, the time-stepped approach will put less strain on the CPU, as despite its higher usage during execution without a display, it will run for a fraction of the time. With a display, the execution is halted between ticks to update the display at a framerate of the user's choosing, and so CPU usage drops. The threaded simulation has no option to disable display updating, but the use of a separate thread for each robot results in a moderate level of CPU usage for a longer period of time.

The lower memory footprint of the time-stepped simulation is most likely due to specific implementation differences. Each robot in the threaded simulation contains a copy of the world map and lists of robots and items, whereas the time-stepped simulation uses a shared resource. While requiring local copies is a factor in any real scenario, it is not required to simulate that unless it is expected that robots will have different local data. If this is a requirement, the memory usage would increase accordingly.

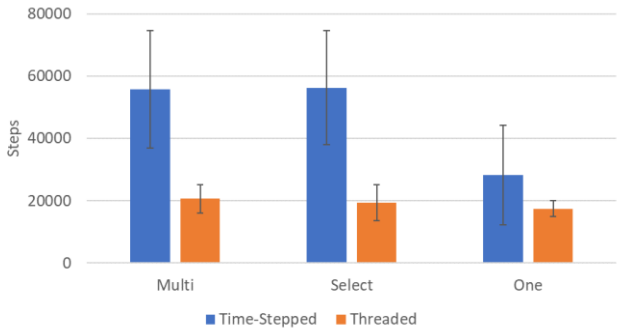


Figure 7. Total steps taken for each simulator using the three strategies in the Equal scenario. Error bars represent one standard deviation.

TABLE II. SIMULATION CPU AND MEMORY USAGE

	CPU (%)	Memory (MB)
Time-Stepped (Live Display)	5-6	30-35
Time-Stepped (No Display)	55-60	30-35
Threaded	35-40	700-750

It can be noted that the stepped approach has the effect of quantizing time, with a tick being the smallest unit possible. This can have the effect of inflating the duration of actions where robots must work in sequence. For example, in the One Responder strategy, it takes 4 ticks to complete the chain of initial broadcast, response, assignment broadcast and eventual robot behaviour change. In this time, another robot could move 4 cells. This has the effect of slowing down that strategy's performance within the simulation. A separate system for dealing with all messages within a single tick may be required to present more accurate results.

The specific implementation details have made comparison of the different strategies between each simulator difficult to achieve, and some of the differences could be removed by re-implementing the previous threaded simulation to adjust the behaviour regarding robots pausing, duplication of data, the requirement for a live display, and the fixed 30x30 map size. This would allow for further comparisons to be made to determine the most suitable approach.

However, as things stand, the quicker execution time and lower impact on the CPU suggests the time-stepped approach is a more favourable system for use as a simulation in the loop for assisting in any adaptation process.

V. CONCLUSION AND FUTURE WORK

The presented research used a time-stepped simulation to investigate the effects of different cooperation strategies for a swarm carrying out a foraging task. It was shown that different situations favour different strategies, with the One Responder strategy proving most effective in a 30x30 map with an equal number of robots, but other strategies providing more reliable performance when faced with an imbalance in the swarm.

Further, different stages of the task will favour different strategies. During the initial phase where large numbers of items remain to be discovered, random exploration with no cooperation strategy produces the best results. Only when a small proportion of the items remain does the adoption of a cooperation strategy start to benefit the performance of a swarm.

Together, these results suggest that giving a swarm the ability to display autonomic adaptive behaviour, adjusting the strategy on the fly based on the current situation, would allow for faster completion of the task.

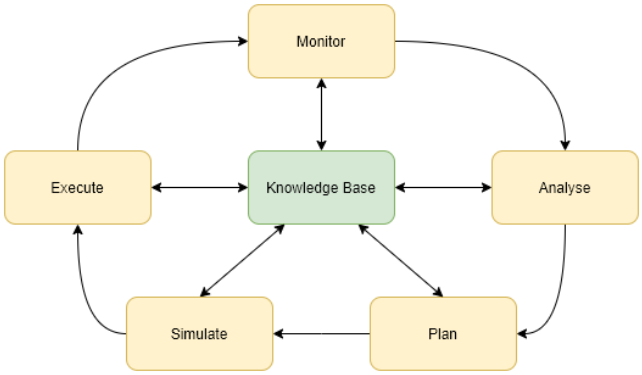


Figure 8. MAPSE-K loop. Simulation is used to test the plans before executing the best performing plan.

The time-stepped simulation was compared against previous work that employed a real-time threaded approach, and was found to have faster execution and reduced load on the host computer. This would make the time-stepped simulation more suitable for use as part of the MAPE-K loop for a foraging swarm, forming MAPSE-K (Figure 8) [1]. This could be achieved by embedding the simulator within one or multiple robots within the swarm, in order to analyse and select strategies without risking reduced performance caused by the trialling of unsuitable candidates in reality.

The expected limited processing capabilities of the host robot mean managing the overhead that simulation entails will become a major factor. A time-stepped approach is thus more suited to this task.

Future work will explore methods of giving the swarm the autonomic ability to adjust its behaviour based on the situation, including the use of simulation-in-the-loop as described. This will include investigating the means by which simulation can be employed in the distributed swarm, allowing for the possibility of incomplete knowledge or local factors influencing the decision as robots in different locations in the field are expected to have different perspectives and experiences. Additional strategies will also be developed that may improve overall performance in this foraging task.

Other factors that may affect performance will need to be considered in future, such as the effects of robot loss during a mission, which will be explored by introducing random failure, including energy costs for actions and message broadcasts that needs to be managed by each robot, or the possibility of unreliable communications that may affect the accuracy of the data in use.

REFERENCES

[1] R. Sterritt et al., 'Inspiration for Space 2.0 from Autonomic-ANTS (Autonomous NanoTechnology Swarms) concept missions', in Proceedings of the 17th BIS Reinventing Space Conference, British Interplanetary Society, Nov 2019

[2] A. Farahani, G. Cabri, and E. Nazemi, 'Self-* properties in collective adaptive systems', in Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct, Heidelberg, Germany, Sep. 2016, pp. 1309–1314.

[3] L. Bayindir, 'A review of swarm robotics tasks', *Neurocomputing*, vol. 172, pp. 292–321, Jan. 2016.

[4] I. Navarro and F. Matía, 'An introduction to swarm robotics', ISRN Robot., vol. 2013, pp. 1–10, 2013.

[5] G. Beni, 'From swarm intelligence to swarm robotics', in Swarm Robotics, Berlin, Heidelberg, 2005, pp. 1–9.

[6] IBM, 'An architectural blueprint for autonomic computing, 4th ed.' IBM White Paper, 2006.

[7] J. O. Kephart and D. M. Chess, 'The vision of autonomic computing', *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[8] R. Sterritt, G. Wilkie, G. Brady, C. Saunders, and M. Doran, 'Autonomic robotics for future space missions', 13th Symposium on Advanced Space Technologies in Robotics and Autonmation (ASTRA 2015) – ESA/ESTEC, Noordwijk, Netherlands, May 2015, European Space Agency, unpublished.

[9] M. Doran, R. Sterritt, and G. Wilkie, 'Autonomic architecture for fault handling in mobile robots', *Innov. Syst. Softw. Eng.*, Apr. 2020, doi:10.1007/s11334-020-00361-8.

[10] M. Torres-Torriti, T. Arredondo, and P. Castillo-Pizarro, 'Survey and comparative study of free simulation software for mobile robots', *Robotica*, vol. 34, no. 4, pp. 791–822, Apr. 2016.

[11] C. Saunders, R. Sterritt, and G. Wilkie, 'Collective communication strategies for space exploration', *J. Br. Interplanet. Soc.*, vol. 72, no. 12, p. 416-430, 2019.

[12] J. Prasetyo, G. De Masi, and E. Ferrante, 'Collective decision making in dynamic environments', *Swarm Intell.*, vol. 13, no. 3, pp. 217–243, Dec. 2019.

[13] J. Zelenka, T. Kasanický, and I. Budinská, 'A self-adapting method for 3D environment exploration inspired by swarm behaviour', in *Advances in Service and Industrial Robotics*, Cham, 2018, pp. 493–502.

[14] N. Capodieci, E. Hart, and G. Cabri, 'An artificial immunology inspired approach to achieving self-expression in collective adaptive systems', *ACM Trans. Auton. Adapt. Syst. TAAS*, vol. 11, no. 2, pp. 6:1–6:25, Jun. 2016.

[15] N. Bredeche, E. Haasdijk, and A. Prieto, 'Embodied evolution in collective robotics: A review', *Front. Robot. AI*, vol. 5, p. 12, 2018.

[16] K. S. Kappel, T. M. Cabreira, J. L. Marins, L. B. de Brisolara, and P. R. Ferreira, 'Strategies for patrolling missions with multiple UAVs', J. Intell. Robot. Syst., vol. 99, pp. 499-515, Sep. 2019.

[17] G. Leu and J. Tang, 'Survivable networks via UAV swarms guided by decentralized real-time evolutionary computation', in *2019 IEEE Congress on Evolutionary Computation (CEC)*, Jun. 2019, pp. 1945–1952.

[18] M. Frasheri, B. Cürüklü, M. Ekström, and A. V. Papadopoulos, 'Adaptive autonomy in a search and rescue scenario', in *2018 IEEE 12th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, Sep. 2018, pp. 150–155.

[19] F. Yan, K. Di, J. Jiang, Y. Jiang, and H. Fan, 'Efficient decision-making for multiagent target searching and occupancy in an unknown environment', *Robot. Auton. Syst.*, vol. 114, pp. 41–56, Apr. 2019.

[20] M. Puviani, G. Cabri, and L. Leonardi, 'Enabling self-expression: The use of roles to dynamically change adaptation patterns', in *2014 IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems Workshops*, Imperial College, London, United Kingdom, Sep. 2014, pp. 14–19.

[21] C. Pinciroli *et al.*, 'ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems', *Swarm Intell*, vol. 6, no. 4, pp. 271-295, Dec. 2012.

[22] N. Jakobi, P. Husbands, and I. Harvey, 'Noise and the reality gap: The use of simulation in evolutionary robotics', in *Advances in Artificial Life*, vol. 929, F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 704–720.

[23] F. Kamrani and R. Ayani, 'Using on-line simulation for adaptive path planning of UAVs', in *11th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'07)*, Oct. 2007, pp. 167–174.

[24] N. Keivan and G. Sibley, 'Realtime simulation-in-the-loop control for agile ground vehicles', *Lect. Notes Comput. Sci. Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinforma.*, vol. 8069 LNAI, pp. 276–287, 2014.

[25] C. Pepper, S. Balakirsky, and C. Scrapper, 'Robot simulation physics validation', in *Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems*, Washington, D.C., Aug. 2007, pp. 97–104.

[26] C. J. E. Castle, N. P. Waterson, E. Pellissier, and S. Le Bail, 'A comparison of grid-based and continuous space pedestrian modelling software: analysis of two UK train stations', in *Pedestrian and Evacuation Dynamics*, Boston, MA, 2011, pp. 433–446.

 ISBN: 978-1-61208-781-8